TITLE: RESOURCE MANAGEMENT APPARATUS, SYSTEMS, AND METHODS
INVENTOR'S NAME: SACHIN DOSHI, ET AL.
SERIAL NO.:   10/750,933   DOCKET NO.:   884.A60US1   REPLACEMENT SHEET

1/4

FIG. 1

TITLE: RESOURCE MANAGEMENT APPARATUS, SYSTEMS, AND METHODS
INVENTOR'S NAME: SACHIN DOSHI, ET AL.
SERIAL NO.:   10/750,933   DOCKET NO.:   884.A60US1   REPLACEMENT SHEET
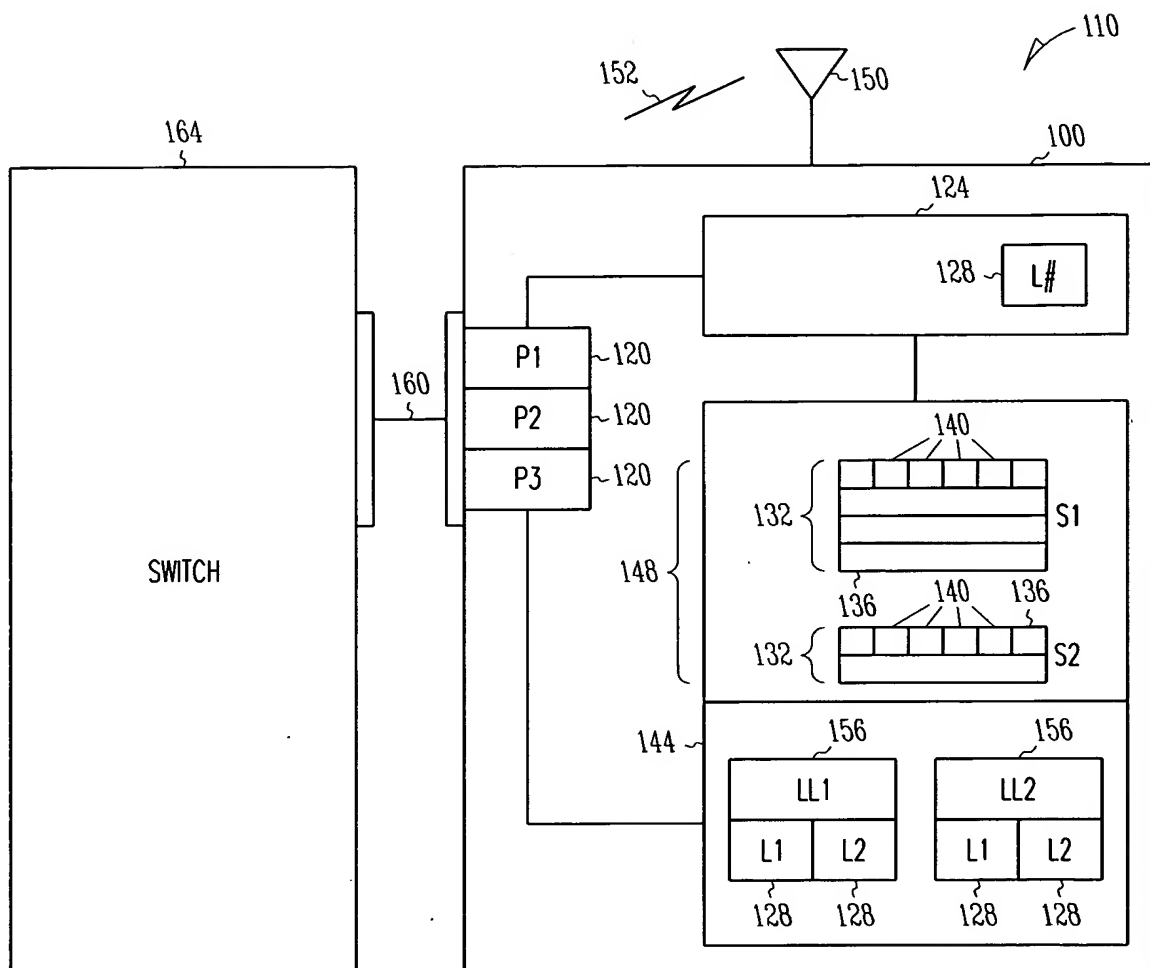
2/4

FIG. 2

TITLE: RESOURCE MANAGEMENT APPARATUS, SYSTEMS, AND METHODS
INVENTOR'S NAME: SACHIN DOSHI, ET AL.
SERIAL NO.:   10/750,933   DOCKET NO.:   884.A60US1

REPLACEMENT SHEET

3/4

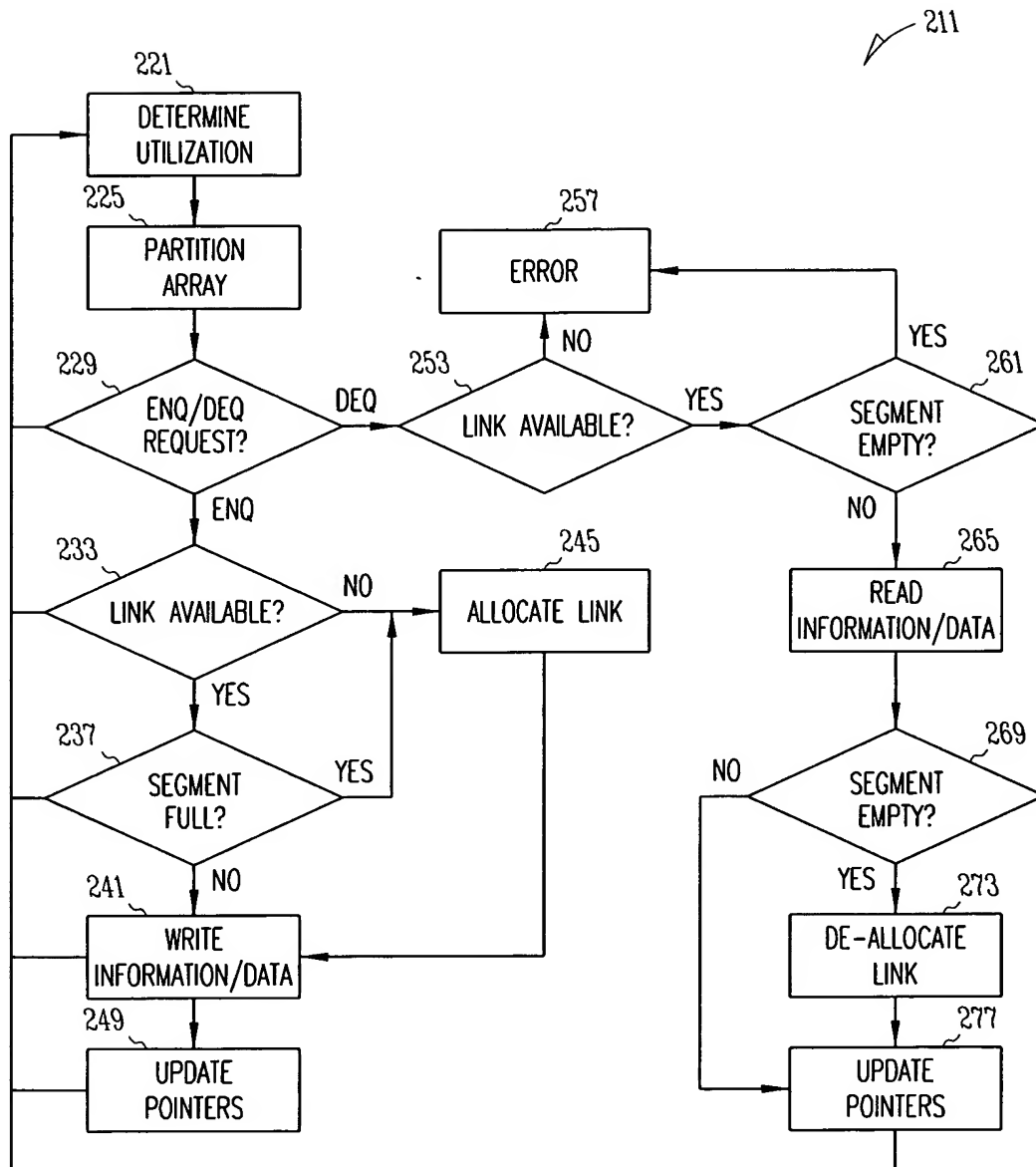```
ENQ_OPERATION
//   This is pseudo code for adding an entry into the link list.
//   WP — Write pointer
//   Info_array: The information array
//   cur_port_id: the ID of the port for which the enqueue operation   ~371
//   Is being done.   Note that many ports are sharing this
//   resource.
//   LR — Link RAM, linked list information stored here
//   This pseudo code is for a 4 to 1 linked list

If   (WP[cur_port_id][1:0]==2'b00){ ~373
//   If the lower 2 bits of the WP of the current port is 0, a new segment
//   is required for this port.
//   Take the free_avail_link that provides a pointer to the first free segment
//   and make the WP point to the next location (01) in the new segment.
//   Store the information in the 1st location (00) of the new segment.
     WP[cur_port_id]<={free_avail_link, 2'b01};
     Info array [{free_avail_link, 2'b00}] <= Info. ~375
}   Else {
//   If the lower 2 bits of the WP are non-zero, it means that the segment has
//   space to store some more information.   Keep adding the information in the
//   empty locations of the segment.   Note that the information storage is
//   sequential within a segment.
     WP[cur_port_id]   <= {WP][cur_port_id][11:2],WP[cur_port_id][1:0]+2'b01}, ~377
     Info_array[WP[cur_port_id]]<= Info;

}

//   This portion keeps track of whether the link list for a port is empty
//   or not.   It also updates the linked list with the new segments.
if   (Empty_Flag[cur_port_id])
          {   if (WP[cur_port_id][1:0]==2'b00)
                    LR[WP[cur_port_id][11:2]]<=free_avail_link; ~379
//   If the linked list for the current port is not empty then whenever
//   a new segment is allocated, link it in the link ram of the current
//   port.
}
Else
//   If the linked list is empty for a particular port then initialize the
//   read and write pointers.   Also reset the empty flag.
//   Note that the read pointer will be updated by the dequeue operation.
          {   Empty_Flag[cur_port_id] <= FALSE. ~381
               if   (WP[cur_port_id][1:0]=2'b00)
                         RP[cur_port_id] <={avail_link_S0, 2'b00};
          }

DEQ_OPERATION
//   this is the pseudo code for the dequeue operation.

Cur_rp = RP[cur_port_id]; ~383
//   generation of the empty condition for the dequeue operation
Empty_condition = WP[cur_port_id]=={Cur_rp[11:2], Cur_rp[1:0]+2'b01}
If (~Empty_condition && Cur_rp[1:0]==2'b11) ~385
//   If the linked list is not empty and we are reading the last location
//   within a segment, then take the read pointer from the link ram.
     RP[cur_port_id] <= {LR[RP[cur_port_id][11:2]], 2'b00};
} else {
//   If we are not reading the last location within a segment then keep on
//   incrementing the read pointer within the segment.
     RP[cur_port_id] <= {RP[cur_port_id][11:2], RP[cur_port_id][1:0] =2'b01}; ~387
}

//   When the segment is completely read, put the free segment in the pool of the
//   free segments.
If(RP[cur_port_Id][1:0]=2'b11) ~389
          Put_free_link (RP[qnum_s0][11:2]);//LR write

//   Set the empty flag whenever the empty condition is detected for a particular
//   port
If (Empty_condition)   Empty_Flag[cur_port_id] = TRUE; ~391
```

*FIG. 3*

TITLE: RESOURCE MANAGEMENT APPARATUS, SYSTEMS, AND METHODS
INVENTOR'S NAME: SACHIN DOSHI, ET AL.
SERIAL NO.: 10/750,933   DOCKET NO.: 884.A60US1
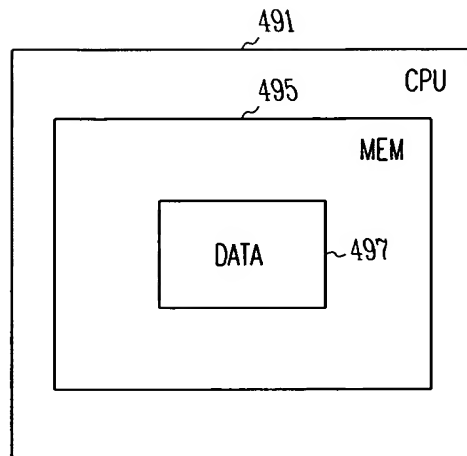
REPLACEMENT SHEET

4/4

FIG. 4